Automatic Game World Generation for Platformer Games Using Genetic Algorithm

Ali Sofyan Kholimi
Informatics Department
Universitas Muhammadiyah Malang
Malang, Indonesia
kholimi@umm.ac.id

Ahmad Hamdani
Informatics Department
Universitas Muhammadiyah Malang
Malang, Indonesia
ahmad_437256@webmail.umm.ac.id

Lailatul Husniah
Informatics Department
Universitas Muhammadiyah Malang
Malang, Indonesia
husniah@umm.ac.id

***Abstract*— Most of the games rely on the game designer to design the level and environment. Increasing of game environment space scale followed by increasing of time and cost. Procedural Content Generation (PCG) is a method to solve this problem by generating a game environment space. In this paper, a PCG method proposed using a genetic algorithm approach to solve the problem in generating game environment. Transition graph adapted in the proposed method to make PCG generate difficulty level. The Index-based approach used to display the biome sequence. This approach displays the biome according to its index in the sequence.**

Keywords*— *Procedural Content Generation, Evolutionary Algorithm, Genetic Algorithm, Transition Graph

I. INTRODUCTION

The game world is an artificial world in which the game event happen. Mostly, a game word is designed by game designer manually. However, the problem will occur if the game world scale is vast. The longer the development of the game, time and cost will increase. While to finish the game does not take a long time [1], The lack of ability to attract a player to play the game again is also a crucial problem [2]. Especially for an indie game developer that has a limited budget and small fans community, it will be an essential problem to solve.

Procedural Content Generation (PCG) is used to solve this problem. PCG is a content that created by algorithm automatically. PCG itself can accept input from user or game designer to adapt the content it will produce [3]. The game that uses PCG will be more varied, give more challenge and have more attractiveness. Open world platformer or exploration platformer has used in this research. The produced game world divided into biome that consists of land, vegetation, and weather. A transition graph that consists of land, vegetation and weather will be built according to game designer design.

The genetic approach has used in generating game level automatically. The genetic algorithm is a method that adapts from the chromosome evolution in the genes of living things[4]. Previous research [5][6][7][8] used a genetic approach to design game level. While in [9] generating game environment based on the biome. Other research [10] in the game world environment has focused on 3D terrain generation. When working on PCG, it is crucial for the game designer to have the ability to control the difficulty curve of the generated level. So, the game designer can create game level according to their design. The, transition graph [11] is adapted and let the game designer design the game world or the level using the graph that included in the genetic algorithm proses. The genetic algorithm will provide result according to the transition graph defined by the game designer. In this research, Transition Graph is used to improve the Genetic Algorithm that used for Automatic Game World Generation for Platformer.

II. RESEARCH METHOD

A. System Overview

In this research, the genetic algorithm is used to design the game world. First, the difficulty curve is defined and include it in a genetic algorithm. When the genetic algorithm finishes its process, the result then passed to the rendering to generate game world based on the design from a genetic algorithm.

B. Difficulty Curve

Difficulty curve using Game World Fitness as a base to control the difficulty of the generated game world. The worse value of Fitness, which is further from zero value, increases the level of difficulty. 30 difficulty values defined for 30 level as shown in Fig 1.

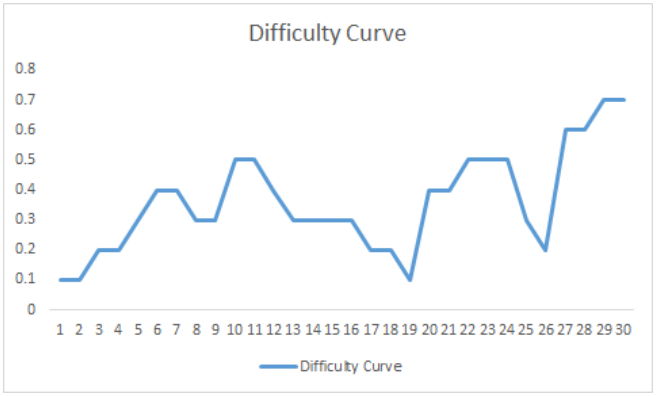

Fig. 1. Difficulty Curve.

C. Generation of Chromosome

The Chromosome in this genetic algorithm consists of a component set that chosen randomly. The Genes for the chromosome contain the sequence of the component shown in Fig. 2. The component itself is a data container that consists of 3 components, lands, vegetation and weather. Game designer determines the type and amount of biome. So, the biome does not limit to the defined one, but it still can be increased.

978-1-5386-8402-3/18/$31.00 ©2018 IEEE

Fig. 2. The sequence of a biome.

D. Fitness Function

Transition graph is used to calculate the fitness value. The transition graph is a graph that consists of lands, vegetation, and weather. Each transition has each own value as given in Fig. 3.

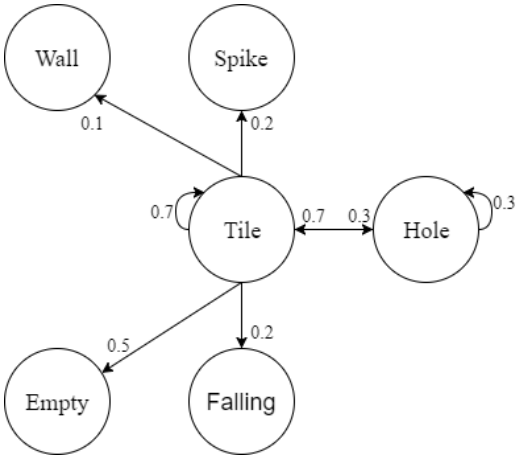

Fig. 3. The transition graph.

Transition graph updated according to the biome used to get better results. The algorithm loops each gene in a chromosome and count the number of chromosomes and divide it by chromosome length calculated by equation (1).

$$d = \frac{numberOfObstacle}{Chromosome\ Length} \quad (1)$$

E. Genetic Algorithm Implementation

The genetic algorithm described in Fig 4. It starts by generating an initial population randomly. The parameter of the genetic algorithm defined. The different problem requires a different parameter value. Our method uses the default parameter that already explained in [12]. Those values are two times of the chromosome size for population, 0.7 for crossover rate and 0.001 for mutation rate. Land data has used for the initial population, vegetation and weather have added in runtime to each chromosome. The genetic algorithm halts when maximum generation reached or more than the threshold. The threshold for halt the algorithm is the difficulty value from the difficulty curve. Another termination condition is counter for counting if the algorithm produces a similar population for a certain amount of generation.

The produced initial population entered a crossover process, and the parent is chosen using roulette wheel selection. A random number from range 0 to the total of the transition value of the population has generated. A chromosome is chosen as a parent if the sum of transition value is less than the random number. Next, genes from the middle until the last index of the second parent are swapped with the first parent genes in the middle until the last index.

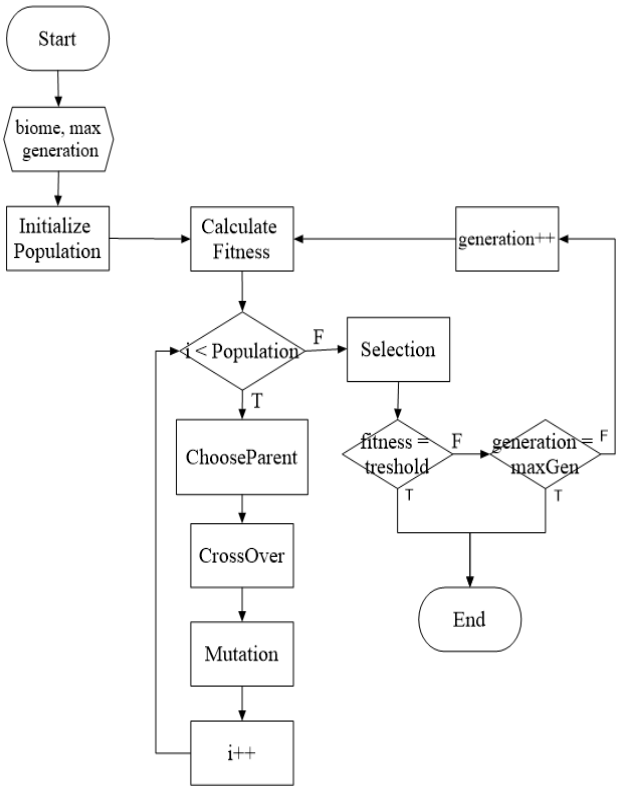

Fig. 4. Genetic algorithm overview.

Child from the crossover process mutate. The probability of mutation happens has decided by mutation rate. There are three mutation operator, that consist of addition, value change, and deletion. It adopts from previous research [11]. A random operator has chosen for the mutation.

1) Value change, choose the value according to the probability in a transition graph.

2) Addition, choose the value to add to the possible value, the value has added to random genes.

Elitism count is used to determine how a chromosome survive to the next generation. It counts N first fittest chromosome in a sorted population that has included in next generation.

F. Graphical Representation

Index-based approach [6] is used to show the biome sequence in the chromosome. This method loops through the chromosome and Instantiate game object according to the component position in the chromosome. The game object that used for instantiation changed according to the game environment type or the design. The land component is Instantiated first followed by vegetation and weather. A random walk is also implemented in the land to make the land looks more natural. The random walk works similar to flipping a coin, if the head has obtained then the y-axis value is increased, otherwise tail decrease the y-axis value while the algorithm walks along an x-axis. Random walk makes the generated land or platform more vary. The random walk also becomes a unique point for generating more game world because the same design shown differently. The result of the graphical representation seen in Fig 5.

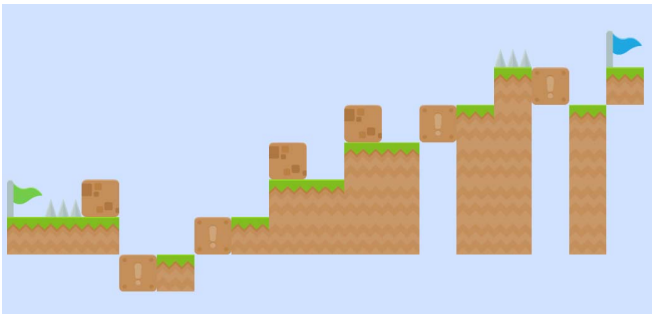

Fig. 5. Generated game world.

III. RESULT AND ANALYSIS

The research aims to generate a game world that corresponding to game designer design and give vary game world variation. The game world graphical representation created based on the generated chromosome from the algorithm. Our proposed method is tested using six scenarios.

A. Scenario 1: Difficulty Curve

Fig. 6 shows the comparison between difficulty curve from designer and the difficulty curve from the generated game world. Root Mean Square Error (RMSE) is also used to compare between difficulty curve from designer and the difficulty curve from our proposed method. RMSE result from our experiment is 0.12.

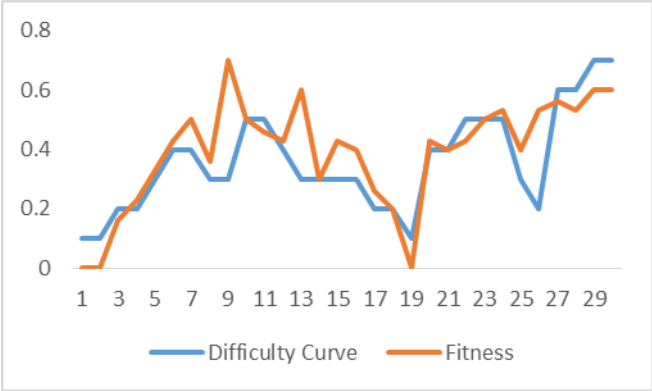

Fig. 6. Comparison between difficulty curve from game designer and generated game world

B. Scenario 2: Playability Testing

In order to know if the generated game world is playable, the generated game world is examined using the following criteria [13].

1) There is a passage connecting the start and endpoint.
2) The passage is fit with the attributes of the player character.

Game played using the generated game world to understand if the generated game world is playable. Table I show that some generated game worlds are not playable although it reaches the given difficulty curve from the game designer. All of the playable game worlds are have fitness under 0.5. However, when difficulty level set higher than 0.6, fitness cannot reach under 0.5.

TABLE I. PLAYABILITY TEST RESULT.

No	Difficulty Level	Fitness	Playtime	Retry	Playable
1	0.1	0	18	0	Yes
2	0.1	0	10	0	Yes
3	0.2	0.16	21	1	Yes
4	0.2	0.23	11	0	Yes
5	0.3	0.33	12	0	Yes
6	0.4	0.43	14	1	Yes
7	0.4	0.5	15	1	Yes
8	0.3	0.36	40	5	Yes
9	0.3	0.7	x	x	No
10	0.5	0.5	18	1	Yes
11	0.5	0.46	23	12	No
12	0.4	0.43	21	4	Yes
13	0.3	0.6	x	x	No
14	0.3	0.3	15	2	Yes
15	0.3	0.43	11	0	Yes
16	0.3	0.4	10	0	Yes
17	0.2	0.26	17	2	Yes
18	0.2	0.2	17	3	Yes
19	0.1	0	8	0	Yes
20	0.4	0.43	19	3	Yes
21	0.4	0.4	53	13	Yes
22	0.5	0.43	9	0	Yes
23	0.5	0.5	x	x	No
24	0.5	0.53	23	3	Yes
25	0.3	0.4	16	0	Yes
26	0.2	0.53	x	x	No
27	0.6	0.56	x	x	No
28	0.6	0.53	x	x	No
29	0.7	0.6	9	0	Yes
30	0.7	0.6	x	x	No

C. Scenario 3: Different Population Size

Population size modified in order to know which values that increase the number of Fitness appearance that has value more than 0.5. Table II shows that population size difference does not have an impact on fitness value.

TABLE II. POPULATION SIZE IMPACT ON THE NUMBER OF FITNESS APPEARANCE THAT HAS VALUE MORE THAN 0.5

Population Size	% of Fitness < 0.5
30	30.3
45	20
60	18.7
75	18.5
90	39.3

D. Scenario 4: Crossover Rate

Crossover rate is value to calculate the possibility of a chromosome to crossover. Better chromosome will bring out better fitness. Table III shows that cross over difference does not have an impact on fitness value.

TABLE III. CROSSOVER RATE IMPACT ON THE NUMBER OF FITNESS APPEARANCE THAT HAS VALUE MORE THAN 0.5.

Crossover Rate	% of Fitness < 0.5
0.5	17.7
0.6	20.5
0.7	15.2
0.8	29.9

E. Scenario 5: Different Mutation Rate

Mutation rate gives a big impact to the genetic algorithm, the higher the value of the mutation rate the algorithm generate more random genes. High mutation rate gives a higher chance to the chromosome in the population to mutate. High mutation rate also gives a negative impact on the fittest chromosome. Because its genes can be mutated and the genetic algorithm loses its fittest chromosome. Table IV shows that the mutation rate difference does not have an impact on fitness value.

TABLE IV. MUTATION RATE IMPACT ON THE NUMBER OF FITNESS APPEARANCE THAT HAS VALUE MORE THAN 0.5.

Mutation Rate	% of Fitness < 0.5
0.001	20.7
0.01	0
0.1	39.8
0.5	8.1
0.25	9.8

IV. CONCLUSION

Our proposed method used for difficulty level under 0.5 based on the difficulty curve. For the next research, our proposed method needs to be improved so it can generate a game world that has a higher difficulty level.

ACKNOWLEDGMENT

This study has supported by the Informatics Department, Universitas Muhammadiyah Malang (UMM) through Internal Research Grant Scheme 2017. Authors are grateful to the Informatics Department, UMM in supporting the present work.

REFERENCES

[1] E. Adams, *Fundamentals of Game Design 2nd Edition*, 2nd ed. New Riders, 2009.

[2] J. Ulisses, R. Gonçalves, and A. Coelho, "Procedural Generation of Maps and Narrative Inclusion for Video Games," *Oporto, January*, no. June, 2015.

[3] N. Shaker, J. Togelius, and M. J. Nelson, *Procedural Content Generation in Games*. Cham: Springer International Publishing, 2016.

[4] D. S. R. Achmad Arwan, "Optimization of Genetic Algorithm Performance Using Naïve Bayes for Basis Path Generation," *Kinetik*, vol. 2, no. 4, pp. 273–282, 2017.

[5] L. Ferreira, L. Pereira, and C. Toledo, "A multi-population genetic algorithm for procedural generation of levels for platform games," *Proc. 2014 Conf. companion Genet. Evol. Comput. companion - GECCO Comp '14*, no. November, pp. 45–46, 2014.

[6] A. B. Moghadam and M. K. Rafsanjani, "A genetic approach in procedural content generation for platformer games level creation," *2nd Conf. Swarm Intell. Evol. Comput. CSIEC 2017 - Proc.*, no. Csiec20 17, pp. 141–146, 2017.

[7] F. Mourato, M. P. dos Santos, and F. Birra, "Automatic level generation for platform video games using genetic algorithms," *Proc. 8th Int. Conf. Adv. Comput. Entertain. Technol. - ACE '11*, p. 1, 2011.

[8] D. F. H. Adrian and S. G. C. Ana Luisa, "An approach to level design using procedural content generation and difficulty curves," *IEEE Conf. Comput. Intell. Games, CIG*, 2013.

[9] Arttu Marttinen, "Procedural Generation of Two-Dimensional Levels," *Metropolia*, 2017.

[10] P. Walsh and P. Gade, "Terrain generation using an Interactive Genetic Algorithm," *IEEE Congr. Evol. Comput.*, pp. 1–7, 2010.

[11] K. Hartsook, A. Zook, S. Das, and M. O. Riedl, "Toward supporting stories with procedurally generated game worlds," *2011 IEEE Conf. Comput. Intell. Games, CIG 2011*, pp. 297–304, 2011.

[12] M. Buckland and A. LaMothe, *AI Techniques for Game Programming*. Premier Press, 2002.

[13] L. A. Ripamonti, M. Mannalà, D. Gadia, and D. Maggiorini, "Procedural content generation for platformers: designing and testing FUN PLEdGE," *Multimedia Tools and Appl*ications, vol. 76, no. 4, pp. 5001–5050, 2017.